Trabalho Prático 1

Introdução à Programação em Assembly

Arquitetura de Computadores

Departamento de Engenharia Eletrónica e de
Telecomunicações e Computadores

3 de março de 2023

1 Objetivos

Este trabalho tem como principais objetivos o exercício da programação em linguagem *assembly* do processador P16, incluindo a organização dos programas em rotinas e a exploração de um ambiente de programação nesta linguagem.

2 Especificação do Exercício

O trabalho consiste no desenvolvimento e teste de um programa para converter para a base 10 quantidades representadas noutras bases de numeração, envolvendo *i*) operações com números inteiros, *ii*) operações com caracteres ASCII, *iii*) utilização de variáveis em memória, *iv*) invocação de rotinas e *v*) acesso a *arrays* em memória.

Na Listagem 1 apresenta-se a especificação do programa pretendido na linguagem C, em que o tipo de dados `char` é utilizado para representar caracteres segundo o padrão ASCII [1], ocupando cada carácter 8 bits com valores possíveis entre 0 e 127, e os restantes tipos de dados utilizados são os definidos na biblioteca C [4].

O programa a desenvolver deverá ser escrito em linguagem *assembly* do P16, respeitando todas as regras da convenção P16 para a utilização de rotinas, e o seu teste deverá ser realizado recorrendo ao simulador do P16.

3 Trabalho a Realizar

1. Considere a seguinte definição da rotina `multiply` que, usando o algoritmo das adições sucessivas, realiza a multiplicação de dois números inteiros sem sinal codificados com 16 bits, em que o registo `R0` recebe o valor do multiplicando e o registo `R1` recebe o valor do multiplicador.

```
multiply:
      mov   r2, #0
while:
      mov   r3, #0
      cmp   r3, r1
      bhs   while_end
      add   r2, r2, r0
      sub   r1, r1, #1
      b     while
while_end:
      mov   r0, r2
      mov   pc, lr
```

   a) Indique, em número de bytes, a quantidade de memória de código ocupada por esta implementação. Justifique a sua resposta.

   b) Comente a seguinte afirmação: *"Para implementar a variável local da rotina, teria sido preferível utilizar o registo `R4` em vez do registo `R2`."*

2. Considere a definição da função `char2nat` que devolve a quantidade representada pelo símbolo `symbol` na base `radix` ou o valor da constante `NAN`, sempre que a base seja superior à 16 ou o símbolo `symbol` não tiver representação na base `radix`. A constante `NAN` corresponde ao maior valor possível de codificar numa variável com tipo `uint16_t`.

   a) Indique, justificando, o valor da constante `NAN`.

   b) Apresente uma definição para a constante `NAN` e indique, justificando, os correspondentes requisitos de memória.

   c) Implemente a rotina `char2nat`. Recomenda-se a elaboração de um programa de teste que permita verificar o comportamento da rotina desenvolvida em diversos cenários de utilização.

```
1  char tst_str0[] = "01011";
2  char tst_str1[] = "709";
3  char tst_str2[] = "9A0F";
4
5  uint16_t tst_results[3] = { 11, 457, 39439 };
6
7  uint8_t error;
8
9  uint16_t char2nat( char symbol, uint16_t radix ) {
10
11      uint16_t number = NAN;
12
13      if ( symbol >= '0' && symbol <= '9' ) {
14          number = symbol - '0';
15      } else if ( symbol >= 'A' && symbol <= 'F' ) {
16          number = symbol - 'A' + 10;
17      }
18      if ( radix > 16 || number >= radix) {
19          number = NAN;
20      }
21      return number;
22 }
23
24 uint16_t str2nat( char numeral[], uint16_t radix ) {
25
26      uint16_t number = 0;
27      int8_t error = 0;
28      uint16_t idx, tmp;
29
30      for ( idx = 0; error == 0 && numeral[idx] != '\0'; idx++ ) {
31          tmp = char2nat( numeral[idx], radix );
32          if ( tmp == NAN ) {
33              number = NAN;
34              error = 1;
35          } else {
36              number = number * radix + tmp;
37          }
38      }
39      return number;
40 }
41
42 int main( void ) {
43
44      error = 0;
45
46      if ( str2nat( tst_str0, 2 ) != tst_results[0] )
47          error |= 1;
48      if ( str2nat( tst_str1, 8 ) != tst_results[1] )
49          error |= 2;
50      if ( str2nat( tst_str2, 16 ) != tst_results[2] )
51          error |= 4;
52
53      return error;
54 }
```

Listagem 1: Descrição em linguagem C do programa a desenvolver.

3. Implemente a rotina `str2nat` que calcula e devolve a quantidade especificada na *string* `numeral`, ou o valor da constante `NAN` em caso de erro. A *string* `numeral` é um numeral cardinal representado na base `radix`.

4. Implemente as definições de todas as variáveis globais apresentadas, definindo as secções necessárias.

5. Implemente a rotina `main` que constitui um teste unitário para a rotina `str2nat`.

4 Avaliação

O trabalho deve ser realizado em grupo e conta para o processo de avaliação da unidade curricular. Cada grupo deverá submeter o trabalho realizado na plataforma Moodle, na forma de listagem do programa desenvolvido (ficheiros .S e .lst), devidamente indentado e sucintamente comentado. As respostas às perguntas formuladas neste enunciado devem ser incluídas na própria listagem do programa, sob a forma de comentários.

A data limite para a entrega dos trabalhos é 27 de março de 2023.

Após esta entrega, o docente responsável pela lecionação das aulas teórico-práticas combinará com cada grupo uma data e hora para a realização da apresentação do trabalho.

Bibliografia

[1] ANSI: *ISO-IR-6: ASCII Graphic character set*, 1975. https://iselpt.sharepoint.com/:b:/s/acp/EUmzOiW_UNpLo4a3F0Z1LRwBDfkDyCFfVUumgRSO4Y9HtA?e=kscsNz, acedido em 03-03-2023.

[2] Dias, Tiago: *Manual de consulta rápida das instruções do P16*. ISEL, Lisboa, Portugal, março 2022. https://iselpt.sharepoint.com/:b:/s/acp/Ect94aOvx4NHnTtZy8fIAVMBex8SHGQErnM4rzYqhOZzcw?e=huQLgy, acedido em 03-03-2023.

[3] Harris, Sarah e David Harris: *Digital Design and Computer Architecture: ARM Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1a edição, 2015, ISBN 978-0128000564.

[4] Loosemore, Sandra, Richard M. Stallman, Roland McGrath, Andrew Oram e Ulrich Drepper: *The GNU C Library Reference Manual*, 2022. https://www.gnu.org/software/libc/manual/html_node/Integers.html, acedido em 03-03-2023.